المدرسة الوطنية العليا في الأمن السيبراني
NATIONAL SCHOOL OF CYBERSECURITY

1ST YEAR BASIC TRAINING IN CYBER SECURITY

INTRODUCTION TO OPERATING SYSTEMS 1 (SYST1)

Dr. Sassi BENTRAD
✉ : sassi.bentrad.enscs@gmail.com | | sassi.bentrad@enscs.edu.dz
LISCO Laboratory (**L**aboratoire d'**I**ngénierie des **S**ystèmes **CO**mplexes) / University of Badji Mokhtar-Annaba (**UBMA**)
National **S**chool of **C**yber **S**ecurity (**NSCS**)

Basic Training in Cyber Security (1BT)

Formation de Base en Cyber-Sécurité (1FB)

Chapter 5

Users, Groups and Permissions Management

SYST1'2025/2026

Module Description Sheet : *Introduction to Operating Systems 1 (SYST1)*

Course Content

User, Group and Managing Permissions (10 %)

CHAPTER 5

- Users, Groups and Permissions
- Permission (Access Mode) by Owner Status
- Superuser (Root User) vs. Normal user
  - ✓ ***sudo*** *(Run Command with Superuser Privileges)*
  - ✓ ***su*** *(Switch User)*
  - ✓ ***useradd*** *(Add User)*
  - ✓ ***passwd*** *(Set Password)*
  - ✓ ***userdel*** *(Delete User)*
- Primary Group and Secondary Group
  - ✓ ***groupadd*** *(Add Group)*
  - ✓ ***usermod*** *(Modify User Account Information)*
  - ✓ ***gpasswd*** *(Add and Delete Users to Group)*
  - ✓ ***groupdel*** *(Delete Group)*
  - ✓ ***chown*** *(Change Owner of File and Directory)*
  - ✓ ***chgrp*** *(Change Group of File and Directory)*
  - ✓ ***chmod*** *(Change Access Mode))*
  - ✓ ***chmod*** *Command with Numbers*
  - ✓ ***w*** *and* ***who*** *(Check Current User Login Status)*
  - ✓ ***id*** *and* ***groups*** *(Check User ID and Group)*
  - ✓ ***getent*** *(Display User and Group Data)*

❖ INTRODUCTION

Linux OS is a multi-users system: this means that multiple users can use the computer simultaneously.

→ **The need for an access permissions policy for directories and files.**

How does Linux manage access permissions to files and directories ?

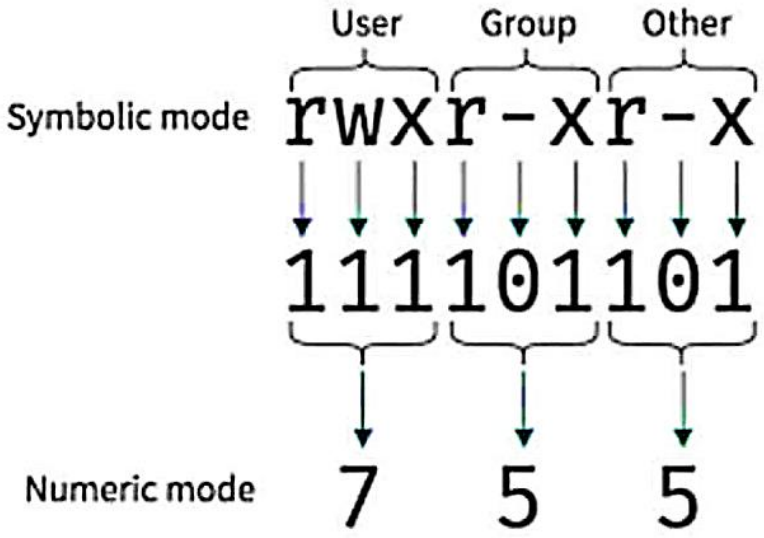
The Linux system assigns to each file and directory **three access permissions**:

- Access permissions for the owner (**User**)
- Access permissions for a group of users (**Group**)
- Access permissions for other users in the system (**Other**)

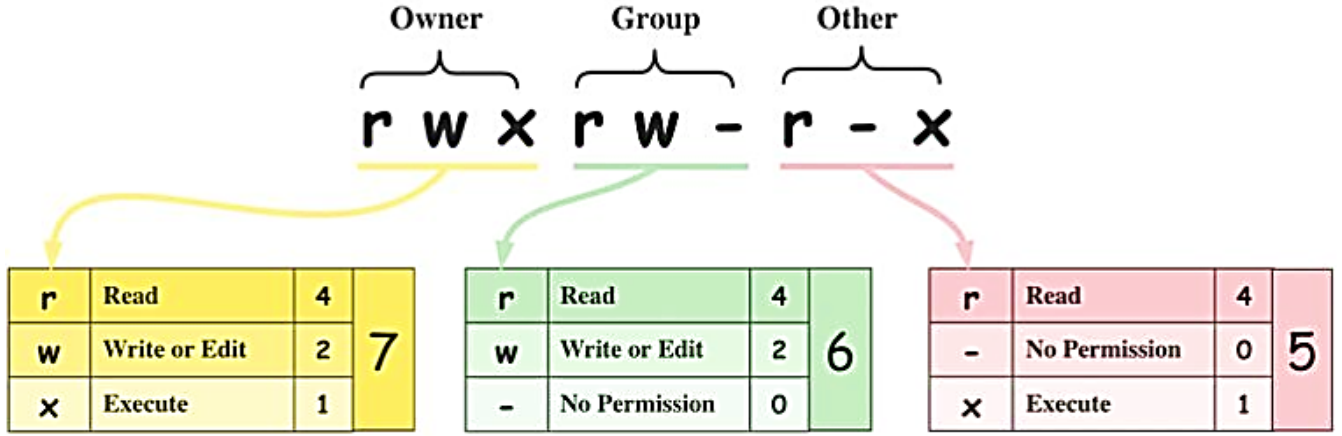
What permissions can be assigned to a file or directory ? Possible permissions

- **r** = **Read** (Read permission)
- **w** = **Write** (Write permission)
- **x** = **Execute** (Permission to execute)
- **-** = **Absence of permission**

❖ LINUX PERMISSIONS



Binary	Octal	String Representation	Permissions
000	0 (0+0+0)	---	No Permission
001	1 (0+0+1)	--x	Execute
010	2 (0+2+0)	-w-	Write
011	3 (0+2+1)	-wx	Write + Execute
100	4 (4+0+0)	r--	Read
101	5 (4+0+1)	r-x	Read + Execute
110	6 (4+2+0)	rw-	Read + Write
111	7 (4+2+1)	rwx	Read + Write + Execute



❖ LINUX PERMISSIONS

Permission Types

- **R** — Read, Can read a file or list the contents of a directory
- **W** — Write, A file or the contents of a directory can be modified
- **X** — Execute, A file can be run as a program or a directory can be entered

User Types

- **U** — User, The owner of the file or directory
- **G** — Group, Other users who are members of the file's group
- **O** — Others, All other users who aren't the owner or part of the group

chown

Changes the owner of a file or directory.

For example:

`chown newowner file`

chgrp

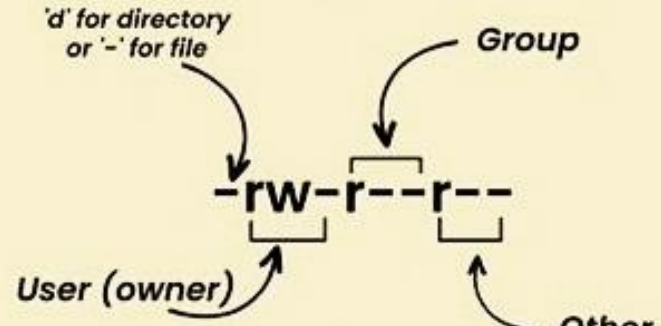
Changes the group of a file or directory

For example:

`chgrp newowner file`

Viewing Permissions

Permissions can be viewed with the ls -l command which returns a string if 10 characters

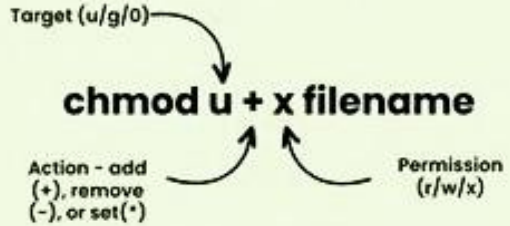


chmod

Changes permissions using symbolic or numeric modes.

symbolic

Defines a target, action, and permissions.

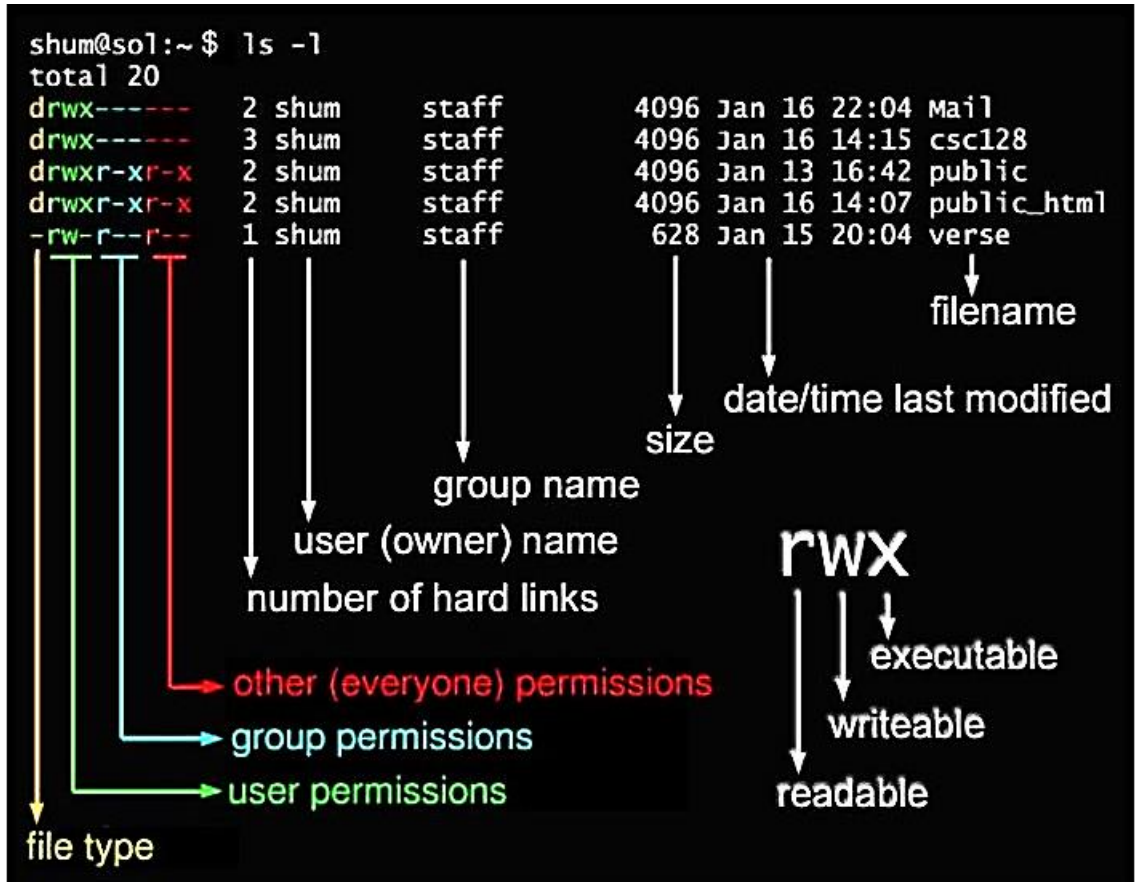


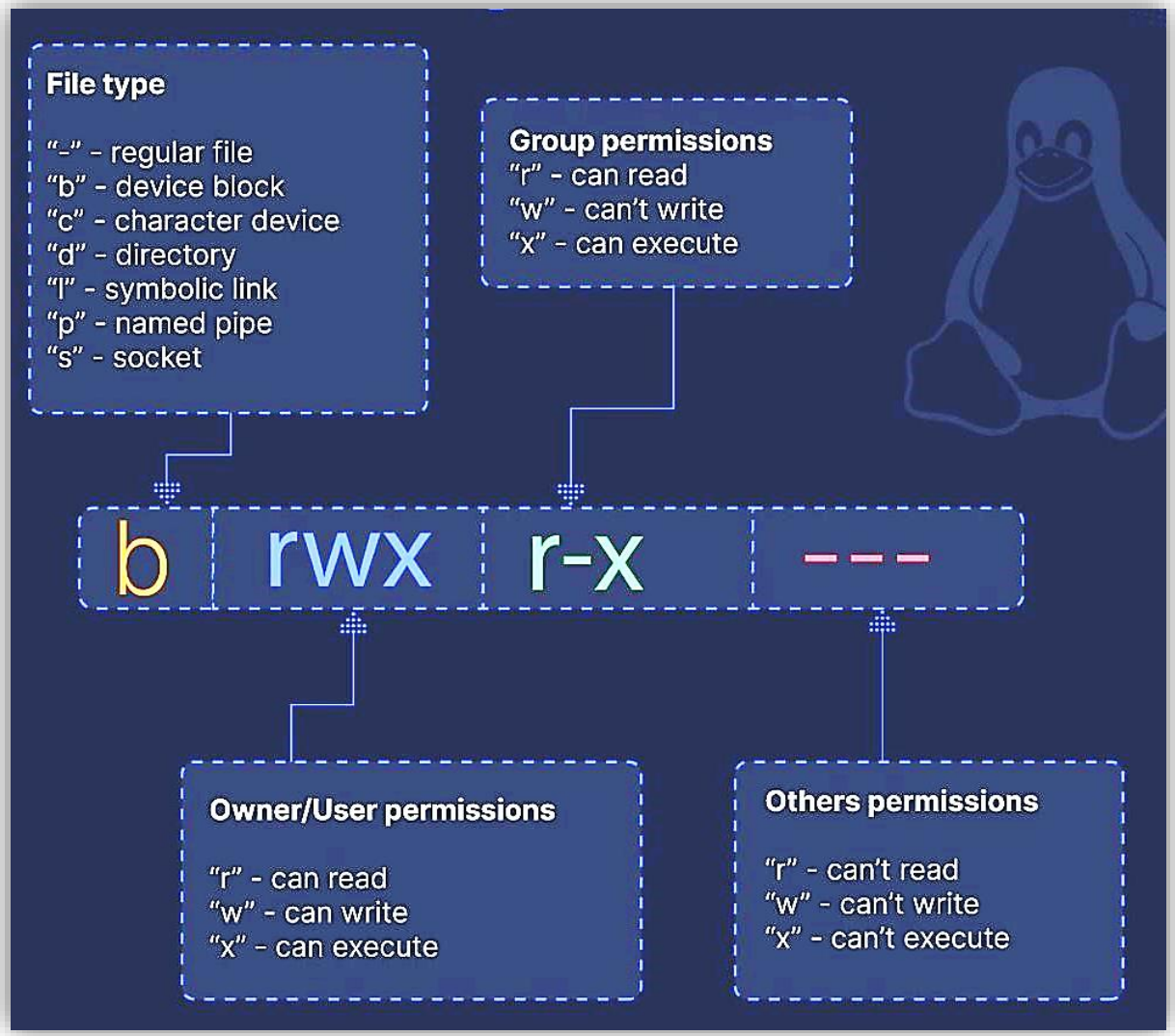
Numeric

Uses the octal numbering system, which uses digits 0 through 7. Read is 4, write is 2, execute is 1, and permission is 0

These numbers are added together to combine permissions. For example, a read and execute permission would be 5 (4+0+1), whereas read and write is 6 (4+2+0)

❖ LINUX PERMISSIONS





" What Are User, Group And Permission in Linux ? "

❑ Superuser (root user) vs. Normal user

In Linux, there are **two types of users** - a **superuser** and a **normal user**.

The **superuser** is a special user account used for system administration.

The **superuser has permissions for all Linux system resources** while **normal users** have limited access to Linux system resources depending on the permission setting for each file and directory.

" What Are User, Group And Permission in Linux ? "

❑ File and Directory owner

Each file or directory has an owner.

Generally, the user who created the file or directory is the owner of the file or directory.

❑ Group

In Linux, you can manage users by group. **File and directory permissions can be more efficient when users are registered in a certain group**.

There are **two types of groups** : **primary group** and **secondary group**.

The **primary group** is used for the **initial owner group** when a new file or directory is created.

" What Are User, Group And Permission in Linux ? "

❑ Owner Status

There are **three owner statuses** of a file or directory :

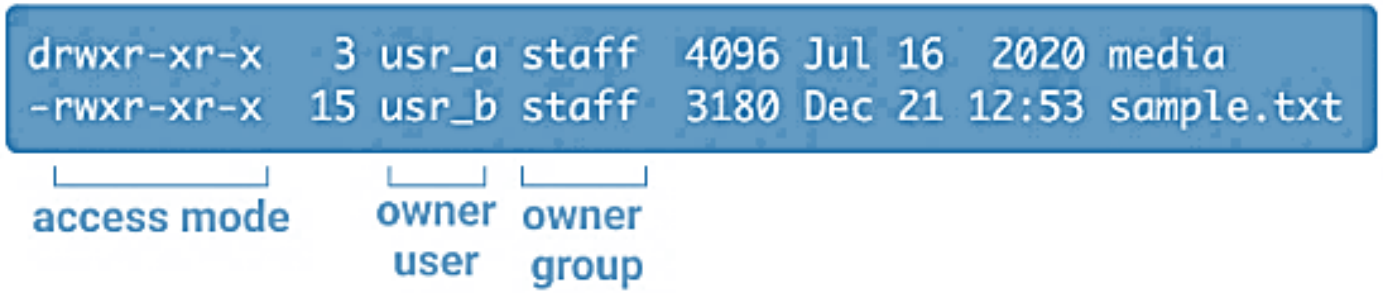
1. **Owner (of the file or directory):** generally, the user who created the file or directory is the owner of the file or directory. You can change the owner by running the **chown** command.
2. **Owner group:** generally, the owner user's primary group becomes the owner group of the file or directory. You can change the owner group by running the **chgrp** command.
3. **Others:** users who are not in the owner group of the file or directory.

" What Are User, Group And Permission in Linux ? "

❑ Permission (Access Mode)

Depending on the **owner status of a file or directory**, the accessibility of the file or directory can be different.

This is called **permission** or **access mode**. When you run the **ls command** with the **-l option**, you can see the access mode of each file by user status.



❖ PERMISSION (ACCESS MODE) BY OWNER STATUS

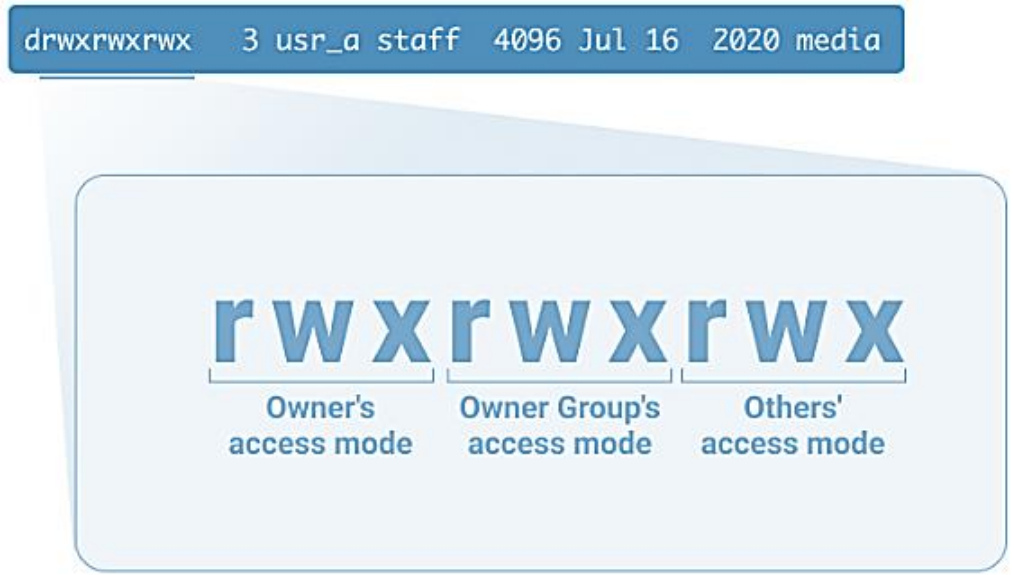
The **ls -l command** gives you the access mode information of a directory or file. Here are the key points of how to read the information.

Three User Types :

Based on the **three owner statuses**, the **access mode** is different for each one.

By running the **ls -l** command, you'll see **9 characters** defining the **access mode**.

- ✓ **The first three letters:** owner's access mode
- ✓ **Next three letters:** owner group's access mode
- ✓ **The last three letters:** others' access mode



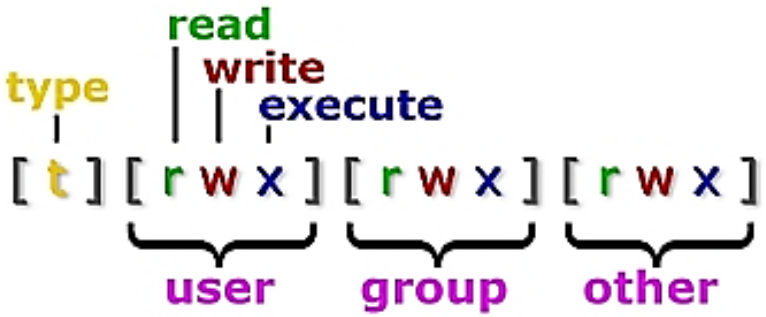
❖ PERMISSION (ACCESS MODE) BY OWNER STATUS

❑ Read (r), Write (w) and Execute (x) permissions

There are three types of **permissions** (**access modes**): **Read** (r), **Write** (w) and **Execute** (x).

The meaning of permissions is **slightly different for files and directories**.

Permission	For File	For Directory
Read (r)	Be able to read the file	Be able to list file and directory names under the directory (read the contents of the directory)
Write (w)	Be able to edit the file	Be able to create and remove files, and change the name of the files in the directory
Execute (x)	Be able to run the command defined in the file	Be able to set the directory as the current working directory (CWD)



❑ Read (r), Write (w) and Execute (x) permissions

Meaning of File Permissions

Files are used to store data so :

- **r = Read :** The contents of the file can be read
- **w = Write :** The content of the file can be modified
- **x = Execute :** execute the file content; Instructions for a binary file or commands for a script.

Attention:

- Permission to delete the file **is not linked** to the file permissions.
- Permission to delete the file is part of the permissions of its directory.

❑ Read (r), Write (w) and Execute (x) permissions

Meaning of Directories Permissions

Directories are used to store files and directories so:

- **r = Read :** The contents of the directory can be read
- **w = Write :** The contents of the directory can be modified, i.e. it is possible to add, create, delete or rename files in the directory
- **x = Execute :** One can run commands on the directory contents.

Attention:

Without **execution** (**x**) permission the directory becomes **locked**, so it is impossible to act on its contents, i.e. neither *access*, nor *add*, nor *create*, nor *delete* or *rename* files or directories in its contents despite the directory bears the permissions to **write** (**w**).

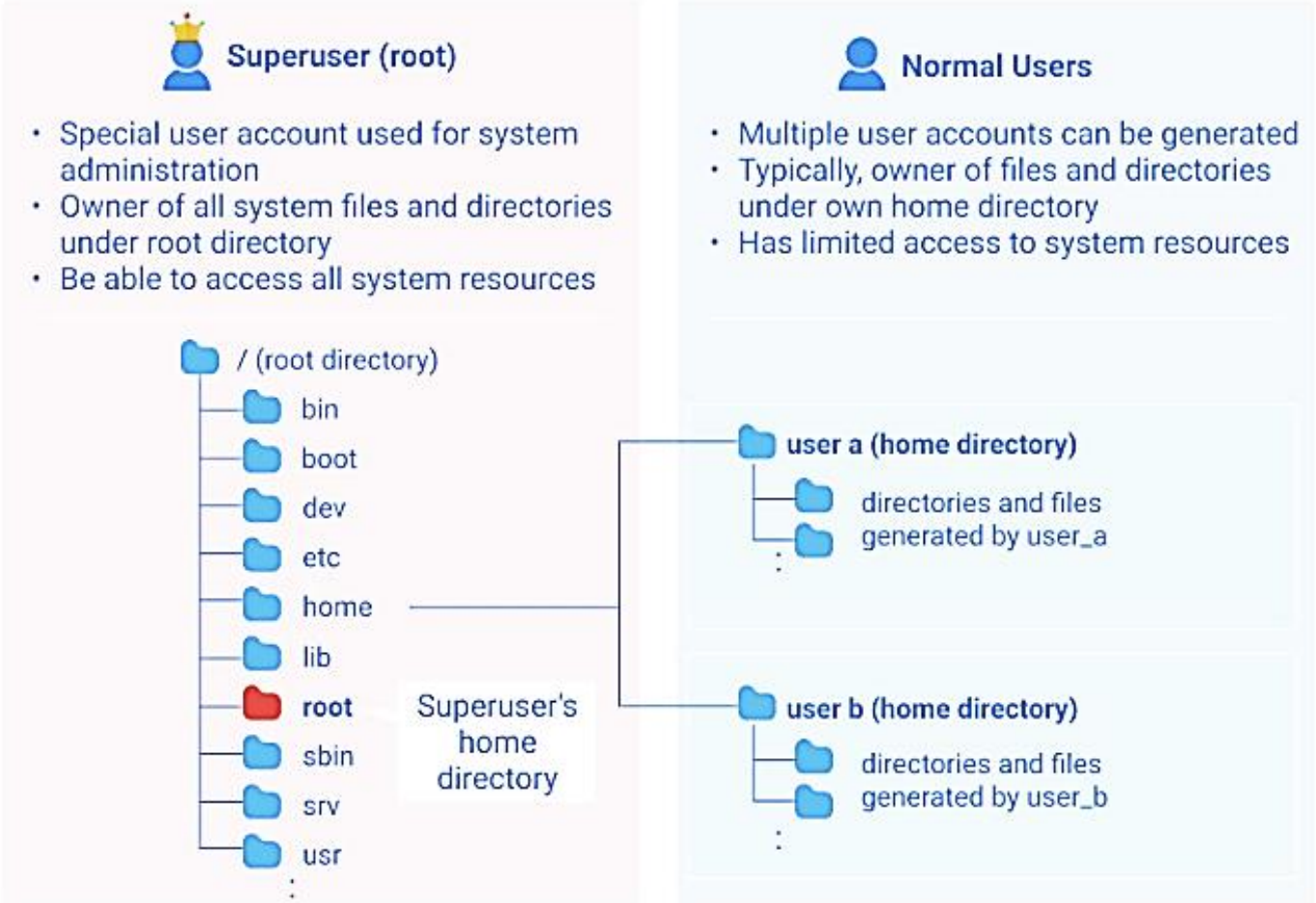
❖ SUPERUSER (ROOT) VS. NORMAL USER

In Linux, there are **two types of users :**

a **superuser** and a **normal user**.

The **superuser** is a special user account used for system administration.

The **superuser** has permissions for all Linux system resources while **normal users** have limited access to Linux system resources **depending on the permission setting for each file and directory**.



LINUX VISUAL GUIDE: Complete beginner's guide
Author: **D-Libro Project** (a lead author: *Ben Bloomfield*)
Edition: First Edition (2024)
D-Libro: https://d-libro.com/course/linux-introduction/

❖ SUPERUSER (ROOT) VS. NORMAL USER

❑ Superuser (Root)

A superuser is a special user account used for system administration. It is also called **root**. As the default setting, the superuser is the owner of all system files and directories under the **/ (root) directory** and has all permissions (**read**, **write**, and **execute**) for all system resources.

❑ Normal Users

Normal users are all users who are not the superuser. **You can create multiple normal users**. Typically, each user has their own **home directory** under the directory path of **"/home"**. Normal users have limited access to directories and files beyond their own home directory.

Example :

- ✓ Although normal users can read several directories and files under the root directory, they cannot overwrite those directories and files.
- ✓ For some directories such as root (root user's home directory) and lost + found, normal users cannot even view directories and files underneath.

❖ SUPERUSER (ROOT) VS. NORMAL USER

❑ sudo (Run Command with Superuser Privileges)

sudo (**S**uper**U**ser **DO**)

Command Line - INPUT

```
ubuntu $ | sudo usermod -aG sudo user_a
```

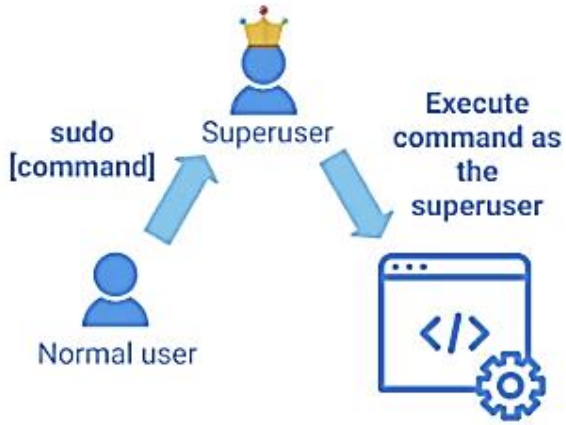
or

Command Line - INPUT

```
ubuntu $ | sudo gpasswd -a user_a sudo
```

[xx] argument

sudo [*command*]



sudoers

- Users who can run the sudo command
- Sudoers are members of **sudo group** on Ubuntu OS (**wheel group** on CentOS)
- To make users sudoers, add them to sudo or wheel group by the **usermod** or **gpasswd** command.

ubuntu — sudo group

CentOS — wheel group

LINUX VISUAL GUIDE: Complete beginner's guide
Author: **D-Libro Project** (a lead author: *Ben Bloomfield*)
Edition: First Edition (2024)
D-Libro: https://d-libro.com/course/linux-introduction/

❑ sudo (Run Command with Superuser Privileges)

What is sudo ?

The **sudo** (**SuperUser DO**) command enables users to run programs with **the security privileges of the superuser**.

Normal users don't have execution permission for some commands, for example, installing a new library.

Logging into the system as the superuser frequently may increase security risks such as password breaches. By using the **sudo** command, you can execute those commands without switching to the superuser.

Who can run the sudo command ? – sudoers

Not all users can execute the **sudo** command. The users who can execute the **sudo** command have to be **members of the sudo group in Ubuntu OS**.

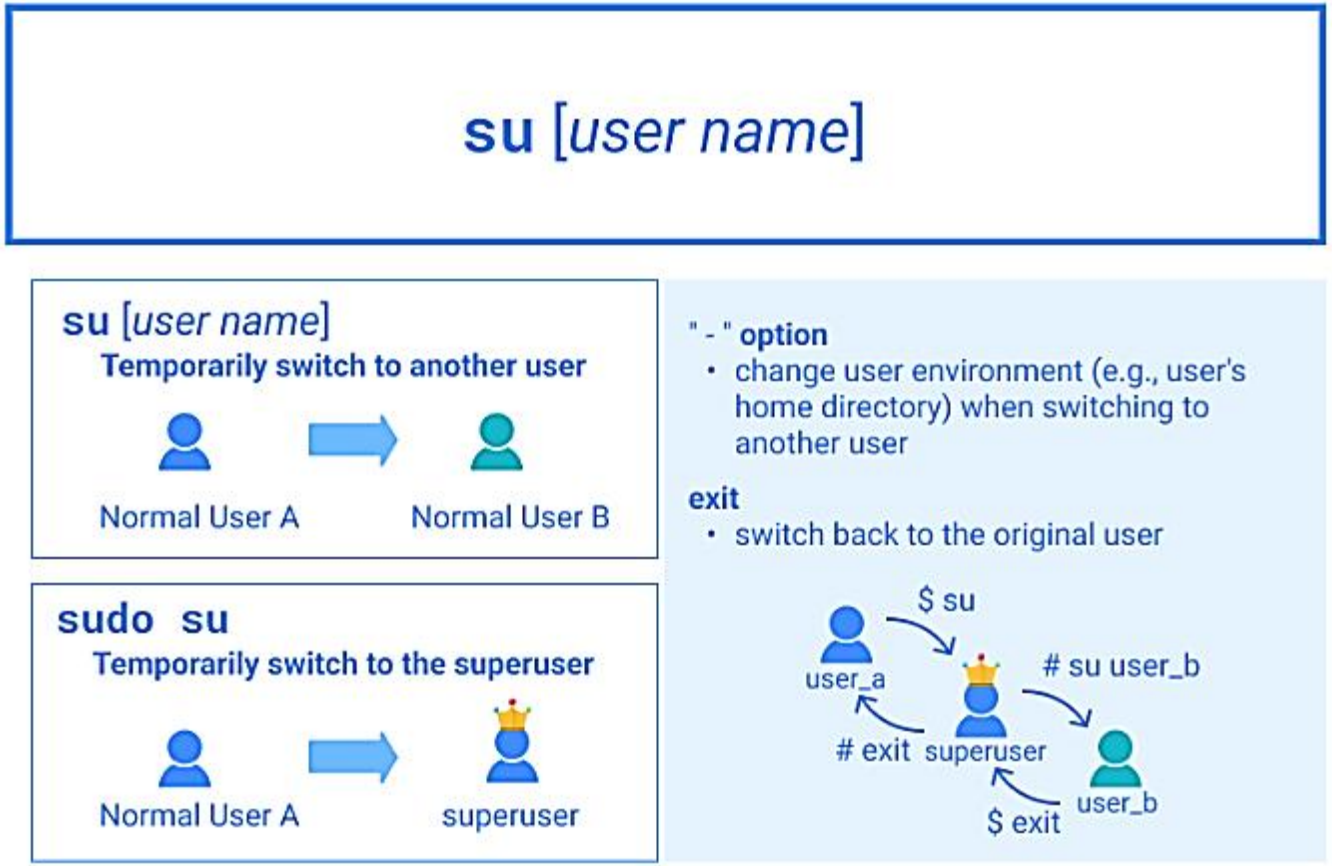
On **Ubuntu**, you can also use the **admin group** instead of the **sudo group**.

❑ su (Switch User)

The **su** command is used to switch to another user temporarily.

You can also **switch to the superuser**.

Switching to the superuser is useful especially when you set up a new server which requires several **new settings** such as adding new users, passwords, groups, and several other system configurations.



LINUX VISUAL GUIDE: Complete beginner's guide
Author: **D-Libro Project** (a lead author: *Ben Bloomfield*)
Edition: First Edition (2024)
D-Libro: https://d-libro.com/course/linux-introduction/

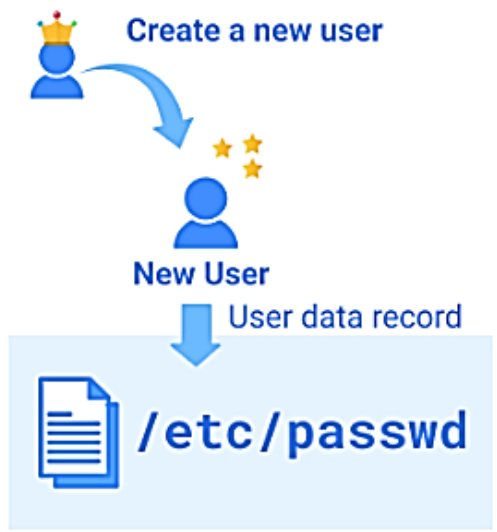
❑ useradd (Add User)

The **useradd** command is used to create a new user.

Usually, the useradd command comes with the **-m** option, which can create the home directory for the new user under the path of **/home** with the user name.

To run this command, you need to run it with superuser privileges.

```
useradd -m [user name]
```



-m option

also, add the home directory for the new user

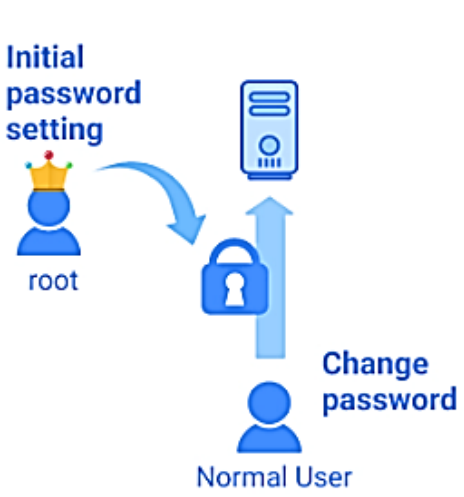
LINUX VISUAL GUIDE: Complete beginner's guide
Author: **D-Libro Project** (a lead author: *Ben Bloomfield*)
Edition: First Edition (2024)
D-Libro: https://d-libro.com/course/linux-introduction/

❑ passwd (Set Password)

The **passwd** (**PASSWorD**) command is used to **set up or change the user password**. When you create a new password, you need to run it with superuser privileges. Once a password is generated, the user can log in to the system.

The user can change the password with this command. With this command, the superuser (or **sudoers**) can control users' login (e.g., lock and unlock the user's account, delete or overwrite the user's password, set password expiry, etc.)

passwd [*user name*]



root's privileges

- delete normal user's password
- overwrite normal user's password
- lock and unlock normal user's account
- manage normal user's password expiry

LINUX VISUAL GUIDE: Complete beginner's guide
Author: **D-Libro Project** (a lead author: *Ben Bloomfield*)
Edition: First Edition (2024)
D-Libro: https://d-libro.com/course/linux-introduction/

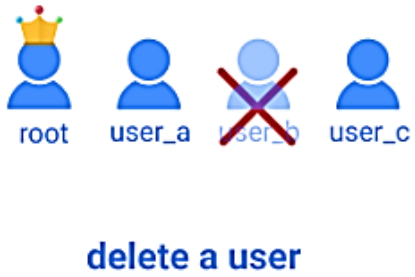
❑ userdel (Delete User)

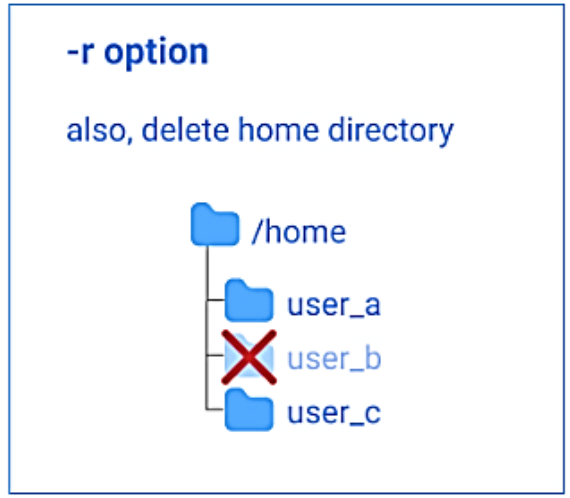
The **userdel** (**USER DELete**) command is used to delete existing users.

With the **-r** option, you can delete its home directory at the same time.

To run this command, you need to run it with superuser privileges.

userdel [*user name*]





LINUX VISUAL GUIDE: Complete beginner's guide
Author: **D-Libro Project** (a lead author: *Ben Bloomfield*)
Edition: First Edition (2024)
D-Libro: https://d-libro.com/course/linux-introduction/

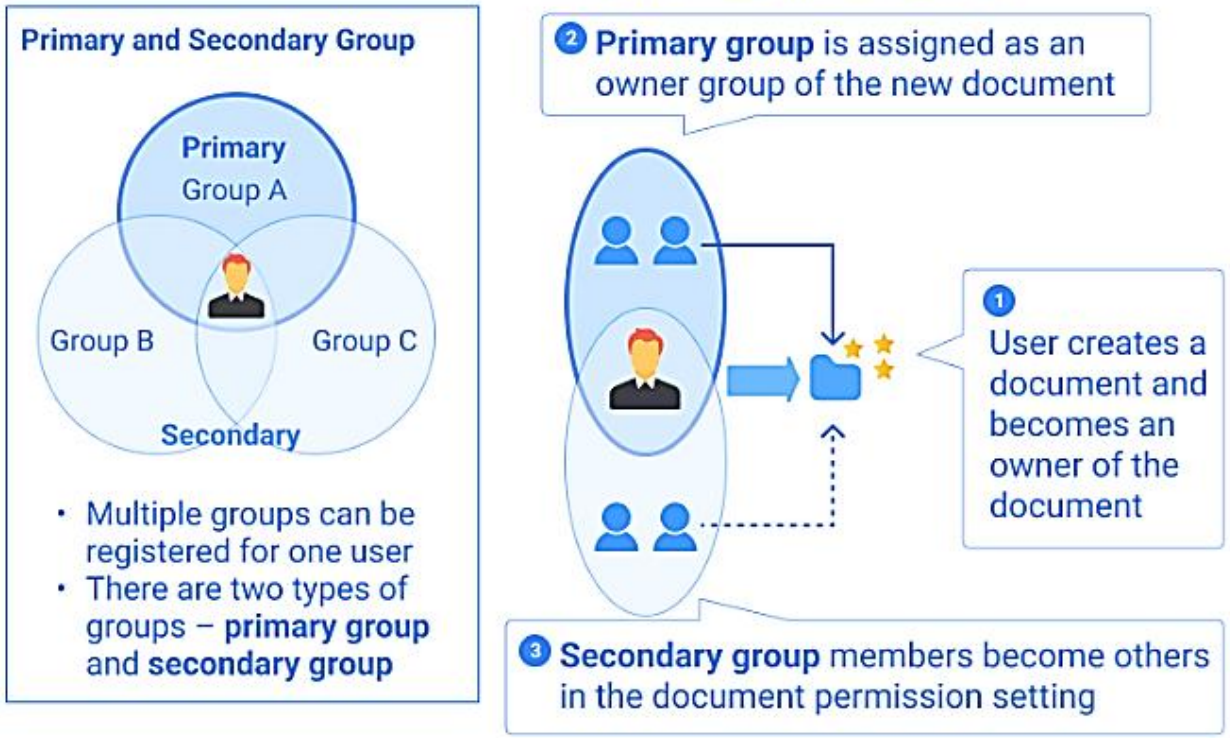
❖ PRIMARY GROUP AND SECONDARY GROUP

Linux OS has a concept of the group to **manage multiple users' permission settings**.

There are two types of groups : **Primary Group** and **Secondary Group**.

❑ Primary Group :

The Primary Group is the main group of the user and it is used for the owner group setting when the user creates a new document. One user can belong to only one Primary Group.



LINUX VISUAL GUIDE: Complete beginner's guide
Author: **D-Libro Project** (a lead author: *Ben Bloomfield*)
Edition: First Edition (2024)
D-Libro: https://d-libro.com/course/linux-introduction/

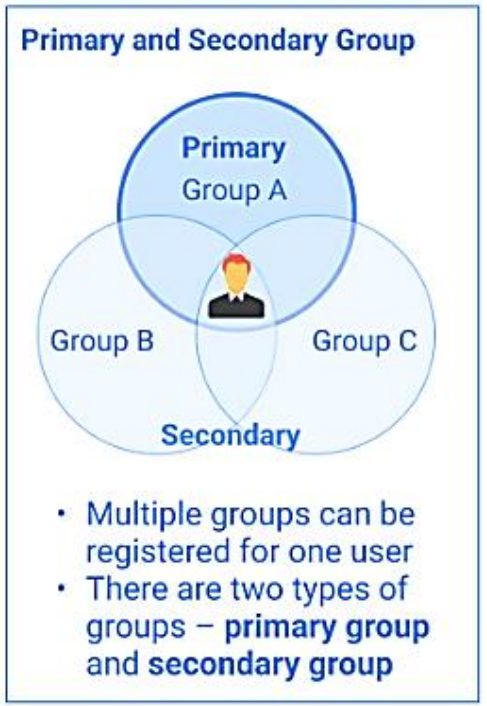
❖ PRIMARY GROUP AND SECONDARY GROUP

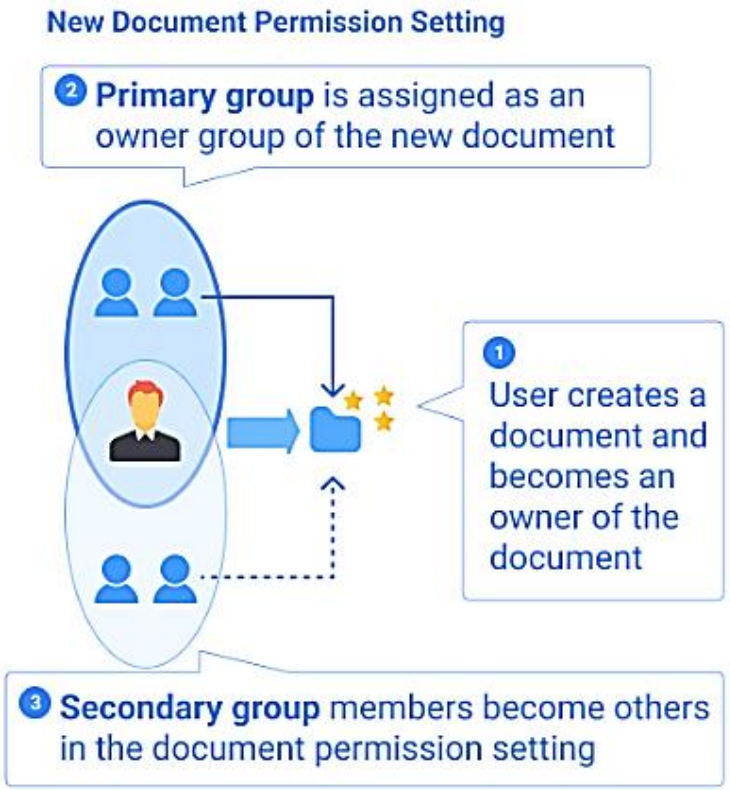
Linux OS has a concept of the group to **manage multiple users' permission settings**.

There are two types of groups : **Primary Group** and **Secondary Group**.

❑ Secondary Group :

- A Secondary Group is used to manage permissions to certain documents (or commands) as a group.
- For example, if user_a is a **sudo** group member on **Ubuntu** (or **wheel** group on CentOS), user_a can run the **sudo** command.
- Unlike in the Primary Group, one user can belong to multiple Secondary Groups.





LINUX VISUAL GUIDE: Complete beginner's guide
Author: **D-Libro Project** (a lead author: *Ben Bloomfield*)
Edition: First Edition (2024)
D-Libro: https://d-libro.com/course/linux-introduction/

❑ groupadd (Add Group)

The **groupadd** command is used to create a new user group.

The group data is recorded under the **/etc/group** file.

To execute this command, you need to run it with superuser privileges.

groupadd [*group name*]

LINUX VISUAL GUIDE: Complete beginner's guide
Author: **D-Libro Project** (a lead author: *Ben Bloomfield*)
Edition: First Edition (2024)
D-Libro: https://d-libro.com/course/linux-introduction/

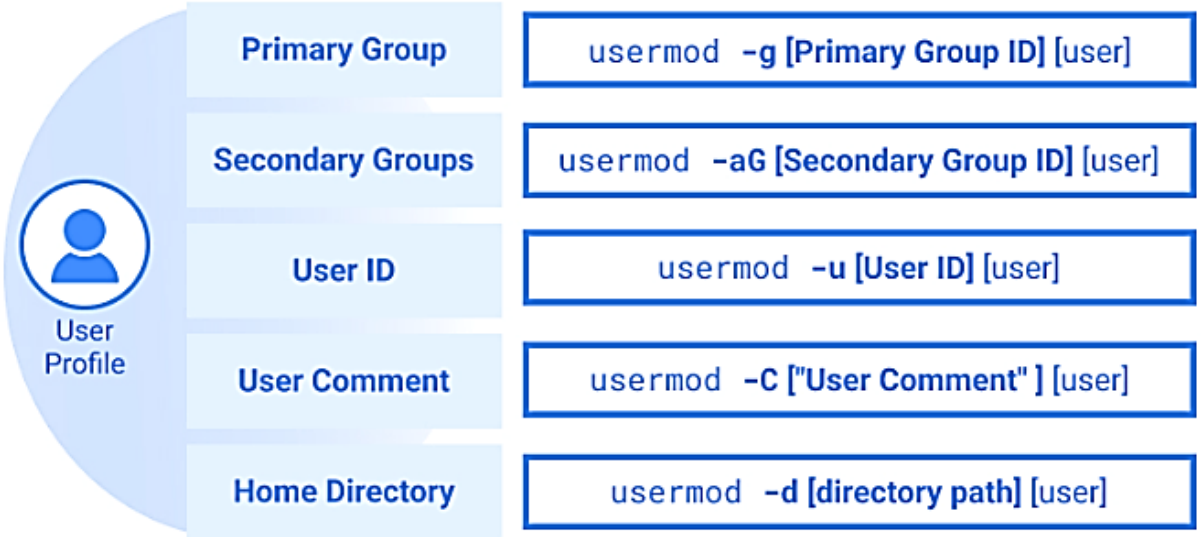
❑ Usermod

(Modify User Account Information)

The **usermod** (**USER MODify**) command is used to modify the user profile.

The user profile includes : *Primary Group, Secondary Groups, user ID, User comment*, and *Home directory path*, etc.

usermod [*option*] [*option argument*] [*user name*]



LINUX VISUAL GUIDE: Complete beginner's guide
Author: **D-Libro Project** (a lead author: *Ben Bloomfield*)
Edition: First Edition (2024)
D-Libro: https://d-libro.com/course/linux-introduction/

❑ gpasswd (Add and Delete Users to Group)

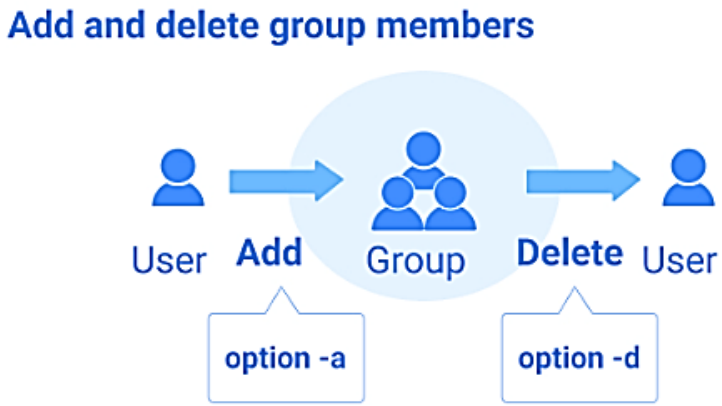
The **gpasswd** command is often used to manage group members.

With the **-a** option, you can add a user to a group. With the -d option, you can delete a user from a group.

With the **-M** option, you can register all group members in a group.

[xx] argument

gpasswd -a *[user name]* *[group name]*
-d *[user name]* *[group name]*



Using this command with **no option**, you can also create a password for a group. As password creation isn't a frequently used functionality of the command, you may find it easy to memorize with the fact that the **gpasswd** command was originally designed to manage the ***/etc/group*** file including group passwords.

LINUX VISUAL GUIDE: Complete beginner's guide
Author: **D-Libro Project** (a lead author: *Ben Bloomfield*)
Edition: First Edition (2024)
D-Libro: https://d-libro.com/course/linux-introduction/

❑ groupdel (GROUP DELete)

The **groupdel** (GROUP DELete) command is used to delete an existing group.

For example, to delete **group_a**, run the following command.

To check **group_a**'s status, check the **/etc/group** file.

You can see that **group_a** no longer exists.

[xx] argument

groupdel [*group name*]

Delete a group

LINUX VISUAL GUIDE: Complete beginner's guide
Author: **D-Libro Project** (a lead author: *Ben Bloomfield*)
Edition: First Edition (2024)
D-Libro: https://d-libro.com/course/linux-introduction/

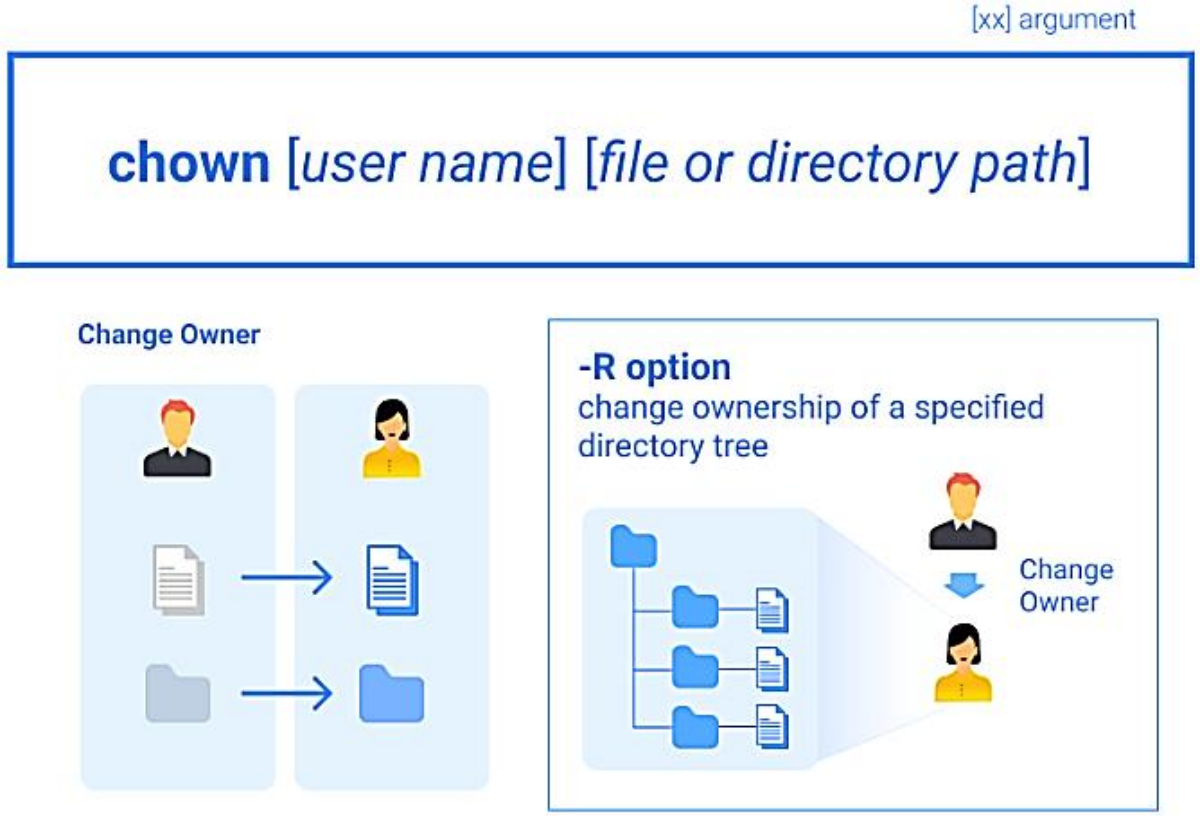
❖ PRIMARY GROUP AND SECONDARY GROUP

❑ chown (CHange OWNer of File and Directory)

The **chown** command is used to change the owner of files or directories.

With the **-R** option, you can change the owner of all files and directories under the specified directory.

To demonstrate the **chown** command, we'll create some directories and a file under the user_a home directory **under superuser privileges**.



LINUX VISUAL GUIDE: Complete beginner's guide
Author: **D-Libro Project** (a lead author: *Ben Bloomfield*)
Edition: First Edition (2024)
D-Libro: https://d-libro.com/course/linux-introduction/

❖ PRIMARY GROUP AND SECONDARY GROUP

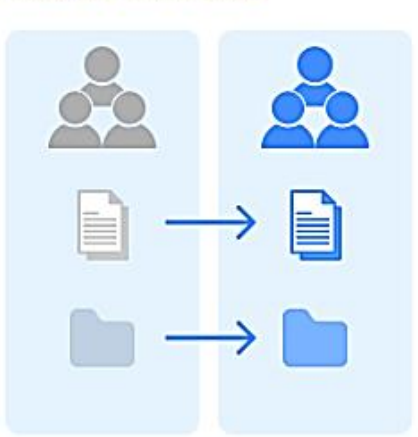
❑ chgrp (CHange GRouP of File and Directory)

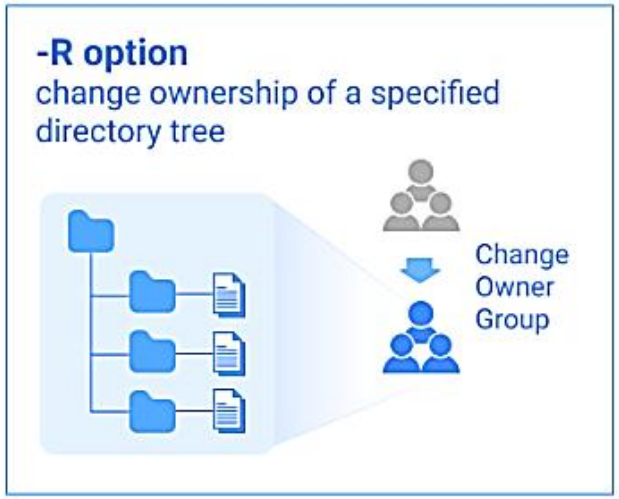
The **chgrp** command is used to change the owner group of files or directories.

With the **-R** option, you can change the owner of all files and directories under the specified directory.

[xx] argument

chgrp [*group name*] [*file or directory path*]

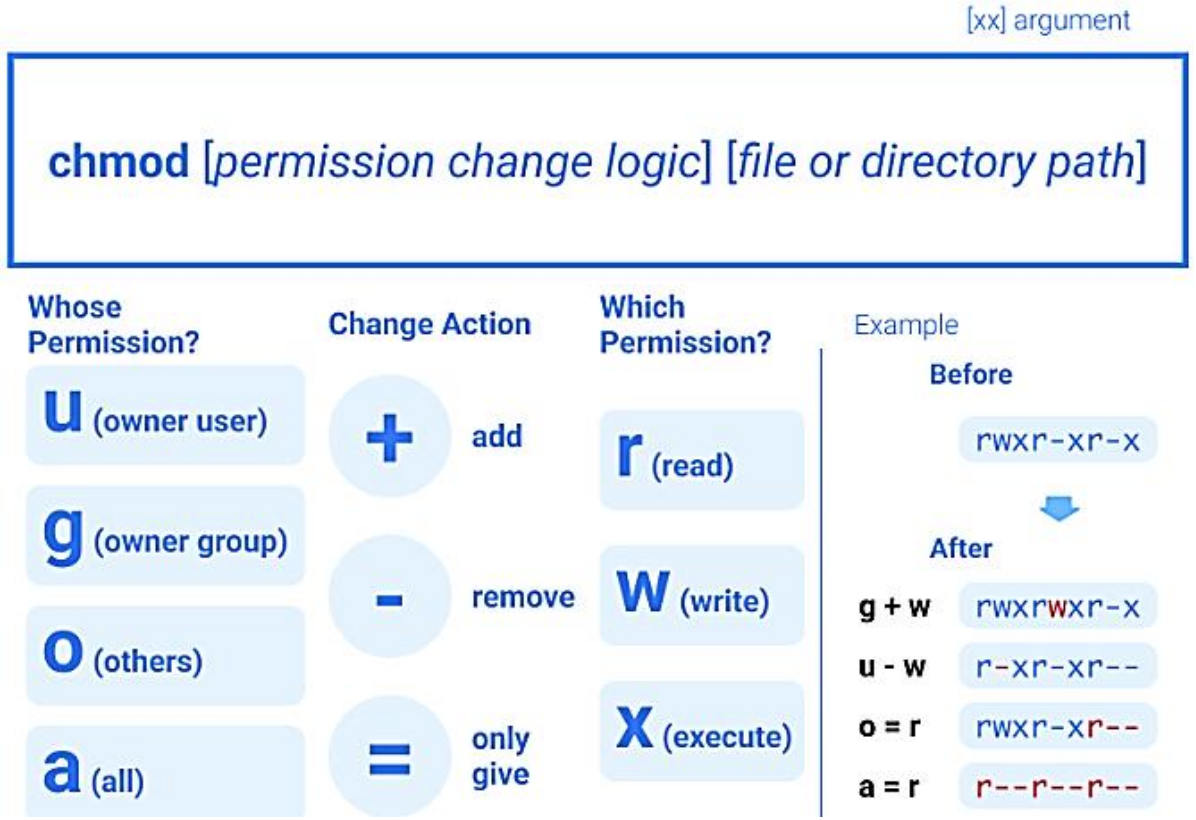




LINUX VISUAL GUIDE: Complete beginner's guide
Author: **D-Libro Project** (a lead author: *Ben Bloomfield*)
Edition: First Edition (2024)
D-Libro: https://d-libro.com/course/linux-introduction/

❑ chmod (CHange Access MODe)

The **chmod** command is used to change the access mode of files and directories.



LINUX VISUAL GUIDE: Complete beginner's guide
Author: **D-Libro Project** (a lead author: *Ben Bloomfield*)
Edition: First Edition (2024)
D-Libro: https://d-libro.com/course/linux-introduction/

❑ chmod (CHange Access MODe)

chmod Command with Numbers

To manage the many combinations and define the access mode of each file or directory in a more efficient way, the **assigned numbers** are also used for the access mode setting.

The numbers are assigned for each permission as shown below.

Using the sum of all numbers, you can represent all access modes with a **single digit for one user type**. To cover the three user types, you need only three digits.

[xx] argument

chmod [*access mode number*] [*file or directory path*]

Permission numbers

Permission	Number
r	4
w	2
x	1
-	0

To define unique access mode, add numbers of each permission

7 (rwx)
6 (rw-)
5 (r-x)
4 (r--)
3 (-wx)
2 (-w-)
1 (--x)
0 (---)

Example:

Owner User	Owner Group	Others
7 (rwx)	4 (r--)	4 (r--)

- Owner user has full permission
- Owner group and others have only read permission

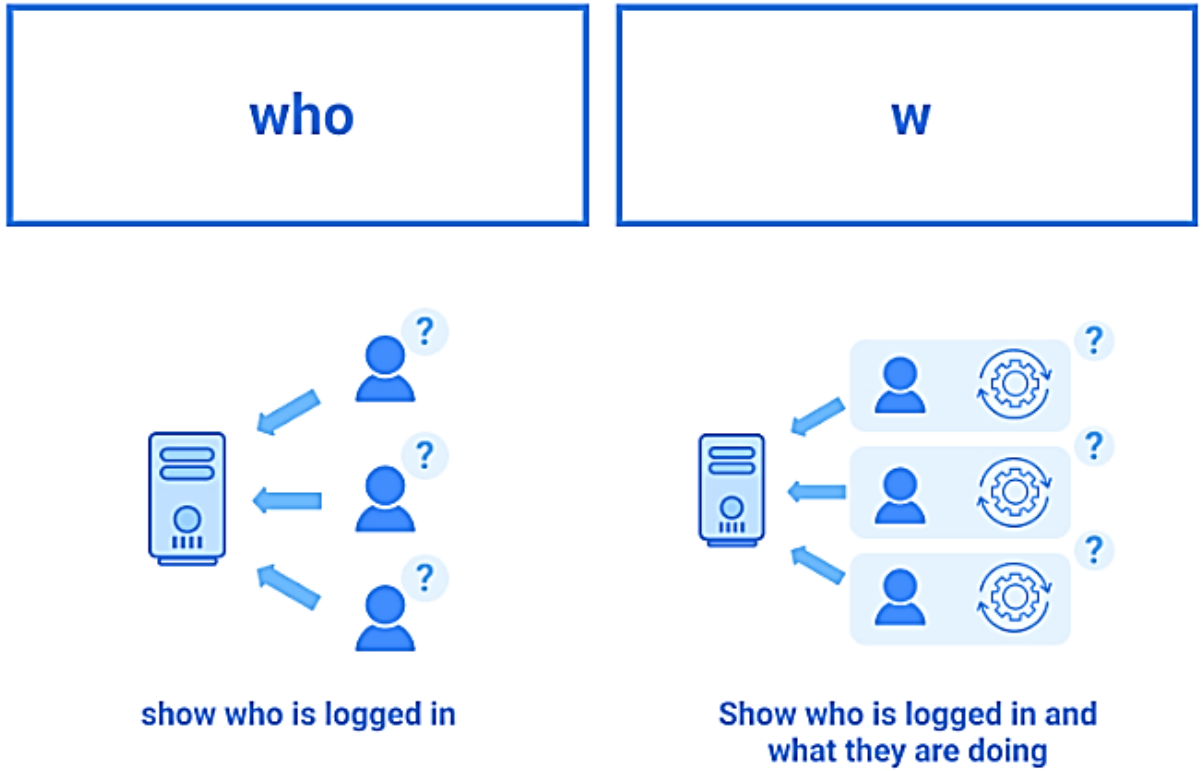
LINUX VISUAL GUIDE: Complete beginner's guide
Author: **D-Libro Project** (a lead author: *Ben Bloomfield*)
Edition: First Edition (2024)
D-Libro: https://d-libro.com/course/linux-introduction/

❑ w and who (Check Current User Login Status)

The **who command** is used to show login records of users currently logged in.

The **w command** is also used to show who is currently logged in but, in addition to that, it also gives information of what they are doing.

These commands don't capture information about the users who are not logged in to the system.



LINUX VISUAL GUIDE: Complete beginner's guide
Author: **D-Libro Project** (a lead author: *Ben Bloomfield*)
Edition: First Edition (2024)
D-Libro: https://d-libro.com/course/linux-introduction/

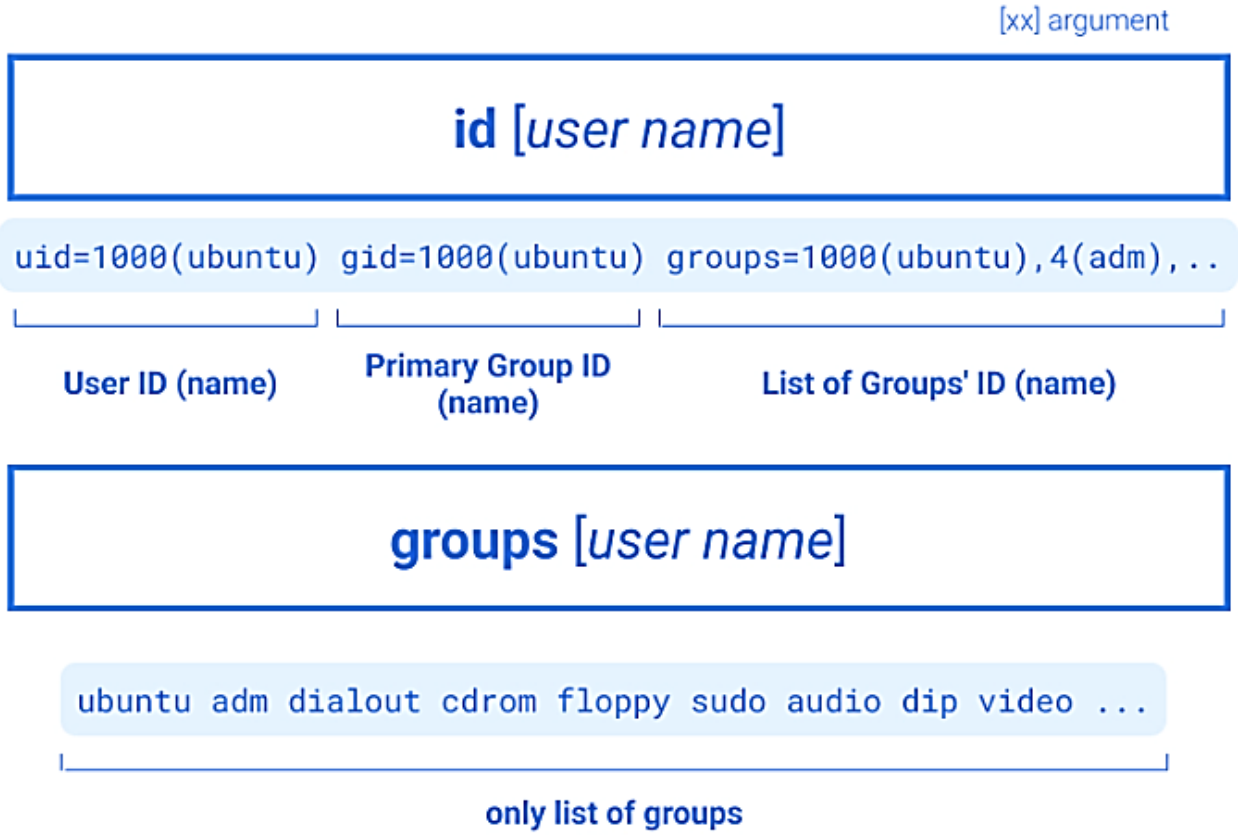
❑ id and groups (Check User ID and Group)

The **id command** returns three types of information.

- User ID
- Primary Group ID and name
- List of Groups' that the user belongs to

The **groups command** returns only the list of groups that the user belongs to.

When you run the command without a user name, the command returns the current user's information. This command is useful to check the latest status when you change a user or group setting.



LINUX VISUAL GUIDE: Complete beginner's guide
Author: **D-Libro Project** (a lead author: *Ben Bloomfield*)
Edition: First Edition (2024)
D-Libro: https://d-libro.com/course/linux-introduction/

❑ getent (Display User and Group Data)

The **getent** command is used to display entries from databases.

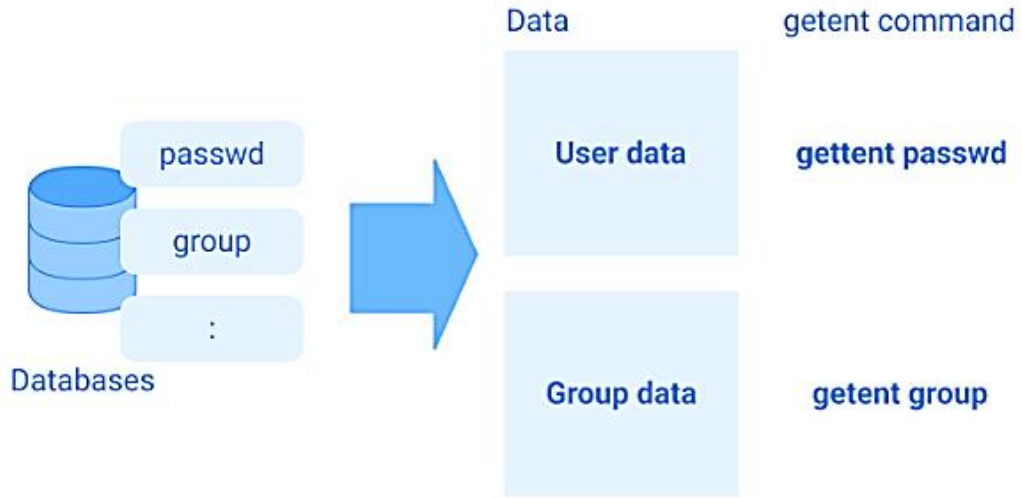
There are multiple databases that Linux OS manages.

For user-related data, there are two major databases : **passwd** for user data and **group** for group data.

The command can also be used to show data from other databases.

[xx] argument

getent passwd [*user name or user ID*]
getent group [*group name or group ID*]



LINUX VISUAL GUIDE: Complete beginner's guide
Author: **D-Libro Project** (a lead author: *Ben Bloomfield*)
Edition: First Edition (2024)
D-Libro: https://d-libro.com/course/linux-introduction/

❑ Modification of permissions

```
$ chmod [options] modifications File/Directory...
```

a. By symbols

The following characters are used:

- **u** To change **user** permissions
- **g** To change **group** permissions
- **o** To change the permissions of **other users**
- **a** To change the permissions of the **user**, **group**, and **other users**.

- **+** To **add** permissions
- **-** To **withdraw** permissions
- **=** To **assign** permissions
- **r, w, x** represent the permissions

❑ Modification of permissions

```
$ chmod [options] modifications File/Directory...
```

a. By symbols

Example:

```
1cpig1b1@ubuntu-virtual-machine:~$ chmod a=rwx Bureau/Cours_TP_linux/
1cpig1b1@ubuntu-virtual-machine:~$ ls -ld Bureau/Cours_TP_linux/
drwxrwxrwx 2 1cpig1b1 1cpig1b1 4096 oct.  10 22:10 Bureau/Cours_TP_linux/
1cpig1b1@ubuntu-virtual-machine:~$ chmod u=rwx,g=x,o=rw Bureau/Cours_TP_linux/
1cpig1b1@ubuntu-virtual-machine:~$ ls -ld Bureau/Cours_TP_linux/
drwx--xrw- 2 1cpig1b1 1cpig1b1 4096 oct.  10 22:10 Bureau/Cours_TP_linux/
1cpig1b1@ubuntu-virtual-machine:~$ chmod o-rw Bureau/Cours_TP_linux/
1cpig1b1@ubuntu-virtual-machine:~$ ls -ld Bureau/Cours_TP_linux/
drwx--x--- 2 1cpig1b1 1cpig1b1 4096 oct.  10 22:10 Bureau/Cours_TP_linux/
1cpig1b1@ubuntu-virtual-machine:~$ chmod g+rw Bureau/Cours_TP_linux/
1cpig1b1@ubuntu-virtual-machine:~$ ls -ld Bureau/Cours_TP_linux/
drwxrwx--- 2 1cpig1b1 1cpig1b1 4096 oct.  10 22:10 Bureau/Cours_TP_linux/
1cpig1b1@ubuntu-virtual-machine:~$
```

❑ Modification of permissions

b. By numerical values

Numerical values in base **8** (**octal**) are used.

- **r** is equivalent to the numerical value **4**
- **w** is equivalent to the numerical value **2**
- **x** is equivalent to the numerical value **1**

Therefore :

- **rwx** is equivalent to the numerical value **7=4+2+1**
- **rw-** is equivalent to the numerical value **6=4+2+0**
- **r-x** is equivalent to the numerical value **5=4+0+1**

❑ Modification of permissions

b. By numerical values

Example:

```
1cpig1b1@ubuntu-virtual-machine:~$ chmod 777 Bureau/Cours_TP_linux/
1cpig1b1@ubuntu-virtual-machine:~$ ls -ld Bureau/Cours_TP_linux/
drwxrwxrwx 2 1cpig1b1 1cpig1b1 4096 oct.  10 22:10 Bureau/Cours_TP_linux/
1cpig1b1@ubuntu-virtual-machine:~$ chmod 716 Bureau/Cours_TP_linux/
1cpig1b1@ubuntu-virtual-machine:~$ ls -ld Bureau/Cours_TP_linux/
drwx--xrw- 2 1cpig1b1 1cpig1b1 4096 oct.  10 22:10 Bureau/Cours_TP_linux/
1cpig1b1@ubuntu-virtual-machine:~$ chmod 710 Bureau/Cours_TP_linux/
1cpig1b1@ubuntu-virtual-machine:~$ ls -ld Bureau/Cours_TP_linux/
drwx--x--- 2 1cpig1b1 1cpig1b1 4096 oct.  10 22:10 Bureau/Cours_TP_linux/
1cpig1b1@ubuntu-virtual-machine:~$ chmod 770 Bureau/Cours_TP_linux/
1cpig1b1@ubuntu-virtual-machine:~$ ls -ld Bureau/Cours_TP_linux/
drwxrwx--- 2 1cpig1b1 1cpig1b1 4096 oct.  10 22:10 Bureau/Cours_TP_linux/
1cpig1b1@ubuntu-virtual-machine:~$
```

How does Linux manage access permissions automatically on files and directories ?

- ✓ **The default permissions assigned to a file: rw- rw- rw- (666)**
- ✓ **The default permissions assigned to a directory: rwx rwx rwx (777)**
- ✓ The system uses a **mask** to control the **default permissions** of files and directories.

- **Calculation of permissions for a file**

  By default : rw- rw- rw- (666)
  Mask : 000 010 010 (022)
  rw- r-- r-- (644)

  So, for the **mask (022)** the permissions assigned to the files automatically are (**666 – 022 = 644**)

- **Calculation of permissions for a directory**

  By defaul : trwx rwx rwx (777)
  Mask : 000 010 010 (022)
  rwx r-x r-x (755)

  So, for the **mask (022)** the rights assigned to the directories automatically are (**777 – 022 = 755**)

How can the mask value be changed ?

The **umask** command is used to control the **mask value**.

- To find out the **numeric value** of the mask : **$ umask**
- To find out the **symbolic value** of the mask: **$ umask -S**
- To change the mask by a **numeric value**: **$ umask** [the numeric value]
- To change the mask by **symbolic value** : **$ umask -S** [the value in symbols]

Example 01:

We want to digitally mask
all permissions attributed to others.

Question:

What is the value of the mask
in numeric format that we will be able to use?

```
1cpig1b1@ubuntu-virtual-machine:~$ umask 007
1cpig1b1@ubuntu-virtual-machine:~$ umask
0007
1cpig1b1@ubuntu-virtual-machine:~$ mkdir Rep
1cpig1b1@ubuntu-virtual-machine:~$ ls -ld Rep/
drwxrwx--- 2 1cpig1b1 g1 4096 oct.  24 09:06 Rep/
1cpig1b1@ubuntu-virtual-machine:~$ touch Fich
1cpig1b1@ubuntu-virtual-machine:~$ ls -l Fich
-rw-rw--- 1 1cpig1b1 g1 0 oct.  24 09:07 Fich
1cpig1b1@ubuntu-virtual-machine:~$
```

Example 02:

We want to automatically mask in **symbolic format** all permissions of the **group** and **others**.

Question:

What is the **symbolic value** of the mask that can be used?

```
1cpig1b1@ubuntu-virtual-machine:~$ umask -S u=rwx,g=,o=
u=rwx,g=,o=
1cpig1b1@ubuntu-virtual-machine:~$ mkdir Rep2
1cpig1b1@ubuntu-virtual-machine:~$ ls -ld Rep2
drwx------ 2 1cpig1b1 g1 4096 oct.  24 09:22 Rep2
1cpig1b1@ubuntu-virtual-machine:~$ touch Fich2
1cpig1b1@ubuntu-virtual-machine:~$ ls -l Fich2
-rw------- 1 1cpig1b1 g1 0 oct.  24 09:23 Fich2
1cpig1b1@ubuntu-virtual-machine:~$
```

❑ Extended Access Permissions (EAP)

How can a user share their file with system users?

Example: A user **user1** wants to share a file **fichier.txt** for users **user2** and **user3** to insert their information.

```
user1@    PC    :~$ chmod o+rw fichier.txt
user1@    PC    :~$ ls -l fichier.txt
-rw-rw-rw- 1 user1 user1 77                    fichier.txt
```

Solution 01:

User **user1** gives **others** (**o**) the **rw** permissions so that **user2** and **user3** can insert their information.

Advantage: Users can insert their information.

```
1
2 Je suis user2 mon telephone number est  660123456 hhhhhhh je suis username user3
3 mon nom est  mohammed, lastname: naime,
4 firstname: bob, lastname: omar, telephone number: 770123456
```

Disadvantage: The file is not protected from false manipulations.

❑ Extended Access Permissions (EAP)

How can a user share their file with system users?

Solution 02:

User **user1** has implemented a program for **user2** and **user3** to use it to insert their information into **fichier.txt**

```
user1@   PC   :~$ ls -l fichier.txt
-rw-rw-rw- 1 user1 user1 0          09:05 fichier.txt
user1@   PC   :~$ ls -l program
-rwxr-xr-x 1 user1 user1 8560       07:08 program
user1@   PC   :~$ su user2
Mot de passe :
user2@   PC   :/home/user1$ ./program
Input your username: user2
Input your firstname: mohammed
Input your lastname: bob
Input your telephone number: 0660123456
user2@   PC   :/home/user1$ su user3
Mot de passe :
user3@   PC   :/home/user1$ ./program
Input your username: user3
Input your firstname: omat
Input your lastname: ali
Input your telephone number: 0550123456
user3@   PC   :/home/user1$ cat fichier.txt
username: user2, firstname: mohammed, lastname: bob, telephone number: 660123456
username: user3, firstname: omat, lastname: ali, telephone number: 550123456
```

❑ Extended Access Permissions (EAP)

How can a user share their file with system users?

Advantage: The program helps **user2** and **user3** to insert their information.

Disadvantage:
The file fichier.txt is still not secure, i.e. a **user2** or **user3** can modify it manually because of the permissions of **others (o) = rw-**

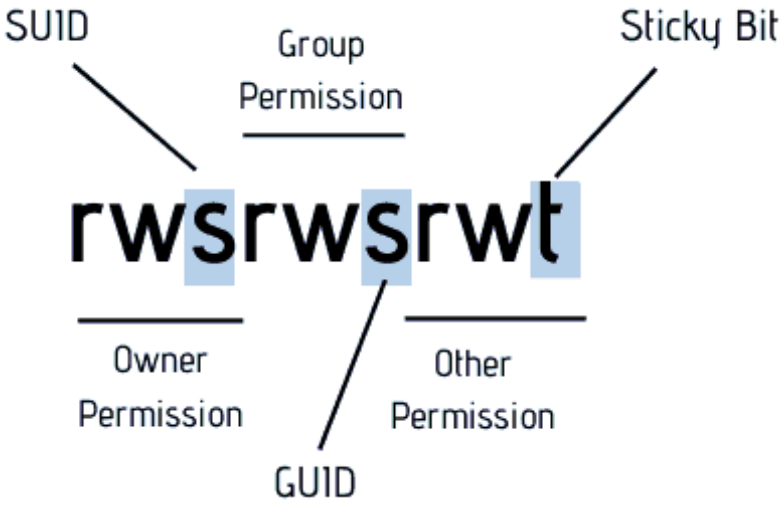
```
user1@ PC :~$ ls -l fichier.txt
-rw-rw-rw- 1 user1 user1 158  09:07 fichier.txt
```

❑ Extended Access Permissions (EAP)

In the Linux system it is possible to establish **extended access permissions** to a **program** (or a **command**) so that it runs with the permissions of the **owner** or the **group**.

Both **SUID** and **SGID** are **special access control bits** in Unix-like operating systems, which affect how programs are executed and how permissions are handled.

These bits are particularly important when a program needs to run with the privileges of **the file's owner** (**SUID**) or **group** (**SGID**), rather than the privileges of the user running the program.



❑ Extended Access Permissions (EAP)

- **In order for a program to run with owner permissions, the SUID bit (Set User ID) is added to the program using : $ chmod u+s program or command**

```
user1@   PC   :~$ chmod o-w fichier.txt
user1@   PC   :~$ ls -l fichier.txt
-rw-rw-r-- 1 user1 user1 0          10:28 fichier.txt
user1@   PC   :~$ chmod u+s program
user1@   PC   :~$ ls -l program
-rwsr-xr-x 1 user1 user1 8560 |        07:08 program
user1@guerrout:~$ su user2
Mot de passe :
user2@   PC   :/home/user1$ ./program
Input your username: user2
Input your firstname: mohammed
Input your lastname: bob
Input your telephone number: 0660123456
user2@guerrout:/home/user1$ su user3
Mot de passe :
user3@   PC   :/home/user1$ ./program
Input your username: user3
Input your firstname: omat
Input your lastname: ali
Input your telephone number: 0550123456
user3@   PC   :/home/user1$ cat fichier.txt
username: user2, firstname: mohammed, lastname: bob, telephone number: 660123456
username: user3, firstname: omat, lastname: ali, telephone number: 550123456
```

❑ Extended Access Permissions (EAP)

- In order for a program to run with group permissions, the GUID bit (Set Group ID) is added to the program using : $ chmod g+s program or command

```
user1@   PC   :~$ chmod o-w fichier.txt
user1@   PC   :~$ ls -l fichier.txt
-rw-rw-r-- 1 user1 user1 158          10:31 fichier.txt
user1@   PC   :~$ chmod g+s program
user1@   PC   :~$ ls -l program
-rwxr-sr-x 1 user1 user1 8560         07:08 program
user1@   PC   :~$ su user2
Mot de passe :
user2@   PC   :/home/user1$ ./program
Input your username: user2
Input your firstname: mohammed
Input your lastname: bob
Input your telephone number: 0660123456
user2@   PC   :/home/user1$ su user3
Mot de passe :
user3@   PC   :/home/user1$ ./program
Input your username: user3
Input your firstname: ali
Input your lastname: omat
Input your telephone number: 0550123456
user3@   PC   :/home/user1$ cat fichier.txt
username: user2, firstname: mohammed, lastname: bob, telephone number: 660123456
username: user3, firstname: ali, lastname: omat, telephone number: 550123456
```

❑ Extended Access Permissions (EAP)

Synthesis: With **extended access permissions**, **SUID** and **GUID bits**, a user can ensure better and more secure sharing of their file.

Example:

A teacher **user1** wants to share a directory **rep/** with his students **user2** and **user3** so that they can put their files: **TP_user2.docx** and **TP_user3.docx**

Solution 01:

The teacher **user1** gives for **others (o)** the **rwx permissions** so that **user2** and **user3** can put their files.

```
user1@_    PC    :~$ chmod o+rw rep/
user1@_    PC    :~$ ls -ld rep/
drwxrwxrwx 2 user1 user1 4096          11:43 rep/
```

❑ Extended Access Permissions (EAP)

Advantage: Users can put their files.

The disadvantage : the user **user3** can delete the file of **user2**.

```
user2@    PC    :~$ cp TP_user2.docx /home/user1/rep/
user2@    PC    :~$ su user3
Mot de passe :
user3@    PC    :/home/user2$ cd
user3@    PC    :~$ cp TP_user3.docx /home/user1/rep/
user3@    PC    :~$ ls -l /home/user1/rep/
total 0
-rw-rw-r-- 1 user2 user2 0          11:55 TP_user2.docx
-rw-rw-r-- 1 user3 user3 0          11:55 TP_user3.docx
user3@    PC    :~$ rm -rf /home/user1/rep/*
user3@    PC    :~$ ls -l /home/user1/rep/
total 0
```

❑ Extended Access Permissions (EAP)

Solution : In a Linux system, it is possible to set **extended access permissions on a shared directory**. So that a user cannot delete only his files even though the directory has **rwx permissions for others (o).**

To establish extended access rights to a directory, the **Sticky bit** is added using the command.

The **Sticky bit** is a special permission that affects the way files are deleted in a **shared directory**. When set on a directory, the **sticky bit** ensures that only the **file owner** or the **root user** can delete or rename files within that directory.

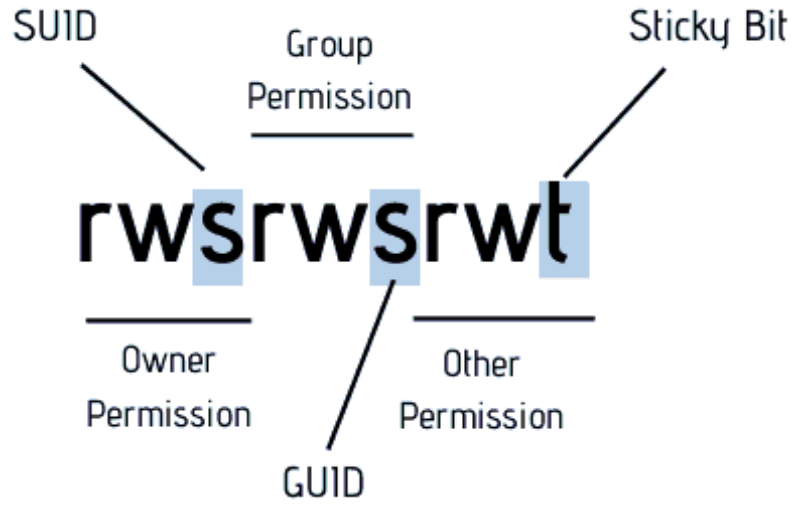
$ chmod o+t directory

```
user1@   PC   :~$ chmod o+t rep/
user1@   PC   :~$ ls -ld rep/
drwxrwxrwt 2 user1 user1 4096          11:56 rep/
user1@   PC   :~$ su user2
Mot de passe :
user2@   PC   :/home/user1$ cd
user2@   PC   :~$ cp TP_user2.docx /home/user1/rep/
user2@   PC   :~$ su user3
Mot de passe :
user3@   PC   :/home/user2$ cd
user3@   PC   :~$ cp TP_user3.docx /home/user1/rep/
user3@   PC   :~$ rm -rf /home/user1/rep/TP_user2.docx
rm: impossible de supprimer '/home/user1/rep/TP_user2.docx': Opération non permise
user3@   PC   :~$ ls -l /home/user1/rep/
total 0
-rw-rw-r-- 1 user2 user2 0                 TP_user2.docx
-rw-rw-r-- 1 user3 user3 0                 TP_user3.docx
```

❑ Extended Access Permissions (EAP)

Extended access permissions in numerical format:

- ❖ The value **4** (**100**) to add the **SUID bit**
- ❖ The value **2** (**010**) to add the **GUID bit**
- ❖ The value **1** (**001**) to add the **Sticky bit**



Example:

```
user1@    PC    :~$ chmod 4755 fichier1
user1@    PC    :~$ chmod 2755 fichier2
user1@    PC    :~$ chmod 1755 repertoire/
user1@    PC    :~$ ls -l fichier1 fichier2
-rwsr-xr-x 1 user1 user1 0          19:22 fichier1
-rwxr-sr-x 1 user1 user1 0          19:22 fichier2
user1@    PC    :~$ ls -ld repertoire/
drwxr-xr-t 2 user1 user1 4096       19:22 repertoire/
```

❑ Extended Access Permissions (EAP)

SUID bit vs. GUID bit vs. Sticky bit

Feature	SUID (`4`)	GUID (`2`)	Sticky (`1`)
Symbol	`s` in `user (u)`	`s` in `group (g)`	`t` in `others (o)`
Affects	Executable files	Executable files & directories	Directories only
Function	Runs with file owner's privileges	Runs with file group's privileges; files inherit group	Prevents non-owners from deleting files
Example	`chmod u+s file`	`chmod g+s dir`	`chmod +t dir`

THANK YOU for your attention!

Questions ?

NSCS
المدرسة الوطنية العليا في الأمن السيبراني
NATIONAL SCHOOL OF CYBERSECURITY

مخبر هندسة الأنظمة المعقدة
LABORATOIRE D'INGÉNIERIE DES SYSTÈMES COMPLEXES

For more information about my research works, **Contact Information**:

Dr. Sassi BENTRAD

LISCO Laboratory : http://lisco.univ-annaba.dz/

☎ : +213 …

✉ : sassi.bentrad.enscs@gmail.com // sassi.bentrad@enscs.edu.dz

in : www.linkedin.com/in/sassi-bentrad/

: http://www.bentrad-sassi.sitew.com/

ORCID
Connecting Research and Researchers

: orcid.org/0000-0002-7458-8121

RESEARCHERID : A-9442-2013

SCOPUS Author ID : 44461052600